L10: Entry 33 of 34 File: TDBD Mar 1, 1997

TDB-ACC-NO: NN9703107

DISCLOSURE TITLE: Heirarchical Data Model for a Relational Database based Geographic Information System

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, March 1997, US

VOLUME NUMBER: 40
ISSUE NUMBER: 3
PAGE NUMBER: 107 - 116

PUBLICATION-DATE: March 1, 1997 (19970301)

CROSS REFERENCE: 0018-8689-40-3-107

DISCLOSURE TEXT:

This document contains drawings, formulas, and/or symbols that will not appear on line. Request hardcopy from ITIRC for complete article. Disclosed is a Relational Data Model for a Geographic Information System (GIS) that supports inheritance. The inheritance model is superior to conventional relational models because it recognizes class relationships that exist in GIS. An implementation of the inheritance model is proposed using relational technology. An actual GIS data model is presented. This model provides support for the three major areas in a GIS: spatial data, application development, and administrative tasks.

First Hit Previous Doc Next Doc Go to Doc#

☐ Generate Collection Print

L10: Entry 33 of 34 File: TDBD Mar 1, 1997

TDB-ACC-NO: NN9703107

DISCLOSURE TITLE: Heirarchical Data Model for a Relational Database based Geographic Information System

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, March 1997, US

VOLUME NUMBER: 40
ISSUE NUMBER: 3
PAGE NUMBER: 107 - 116

PUBLICATION-DATE: March 1, 1997 (19970301)

CROSS REFERENCE: 0018-8689-40-3-107

DISCLOSURE TEXT:

This document contains drawings, formulas, and/or symbols that will not appear on line. Request hardcopy from ITIRC for complete article. Disclosed is a Relational Data Model for a Geographic Information System (GIS) that supports inheritance. The inheritance model is superior to conventional relational models because it recognizes class relationships that exist in GIS. An implementation of the inheritance model is proposed using relational technology. An actual GIS data model is presented. This model provides support for the three major areas in a GIS: spatial data, application development, and administrative tasks.

The evolution of GIS, Computer Aided Design (CAD), and Computer Aided Manufacturing (CAM) has required that information be stored and retrieved from non-proprietary databases within an enterprise. This information is very complex. In particular, the commonality between basic tables that model real world objects in GIS/CAD/CAM is difficult to represent in most relational databases. For a GIS, examples of objects include: o A road o A political boundary o A user interface o A user-defined table o System administrative data One of the main obstacles to modelling these tables with relational technology is the ability to recognize that elements of each table are common across many tables. The trend for modelling this commonality has been the adoption of object-oriented databases and software by many GIS/CAD/CAM systems. Businesses have not benefitted from this direction, however, as most enterprises use relational databases. The problem with relational databases is that their table/view mechanism does not provide the flexibility to define a single view across several tables without a relational join. Consider the following: o A road table has "ID", "TYPE", "LOCATION", and "LENGTH" attributes. o A political boundary table has "ID", "TYPE", "LOCATION", "PERIMETER" and "AREA" attributes. What is proposed is: o An Inheritance Data Model for a GIS.

o A mechanism for storage/retrieval of said data model using a relational database. The following represents the basic description of the inheritance data model used in GIS: General Concepts GIS data includes all the enterprise assets (such as parcels, cables, valves, or highway sections) that are modelled and maintained as GIS data, as well as application and operational data (e.g., Macros, Display Rules, User Profiles, and Device Characteristics) used by the GIS software and GIS applications.

For example, an enterprise may choose to model and maintain land parcel information as GIS data in a GIS. The GIS users and application developers can operate on parcels and the attributes of parcels much as they would access other data in the relational database. In addition, non-GIS users and Corporate applications can access the parcel data through the relational views and application programming interface provided by a GIS.

In a GIS, to determine the appraised value of the parcel owned by John Smith, an user could issue the following query: Select Appraised_Value from Parcel where Owner = 'John Smith' In this example, 'Appraised_Value' and 'Owner' were defined by the application developer or database designer as attributes of Parcel data in the GIS. These attributes are called user defined attributes in the GIS data model. The GIS will also maintain one or more system attributes for each type of GIS data. System attributes are accessed in the same fashion as user-defined attributes in the GIS, for instance, to determine the value (maintained by the GIS) for the area of the parcel owned by John Smith, the following query could be used: Select Area from Parcel where Owner = 'John Smith' Application and Data Independence Under the proposed GIS, the Parcel data discussed in these examples would be physically stored in a relational database. The user-defined attributes, system attributes, and relationships comprising a single Parcel will be distributed among more than one physical table in the relational database. However, the query in the example above is independent of how the Parcel data is physically stored; the user simply operates on Parcel. The concept of application independence from physical storage is important, since it protects applications from the changes to the physical storage format (physical tables). As these changes are made, to provide performance enhancements or to take advantage of enhancements to the underlying relational database, customer applications can be protected. The concept of Views in the relational database is an example of a means of providing application and data independence. A relational view presents a single 'array' of columns that can be operated on in the same fashion as a single physical table, but the columns of the view can represent data that reside in several physical tables, and a column of the view can contain calculated data that is not formally stored in any physical table. Relational applications operating on that view can be protected from changes to the physical tables, in that if the physical table data is re-arranged, the view can be re-created to accommodate the change without having to re-write customer applications that access the view.

GIS Data Model Concepts Relational Views and GIS Object Views In the GIS, an user or application developer can operate on Parcel data much as he would a relational view, although the underlying data for the Parcel is not physically accessed through a single relational table or view. Different components of the parcel data are mapped to a set of columns, as in the case of user and system defined attributes; other components are more complex, such as: picture path elements, edge associations and boundary associations. It is important to note that although the underlying data model for the user's Parcel data can not be mapped to a single set of columns, the GIS permits user access to the Parcel data in a relational fashion as shown in the previous examples. This is done through GIS Object Views. In the GIS data model, an Object View is analogous to a relational view. Users and application developers operate on GIS Object Views much as they would relational views. Object Views provide the data independence for GIS applications that views provide for relational tables. In the previous examples, Parcel is an example of a user-defined Object View created for the storage and maintenance of the enterprise's land parcel data. Several object views can be provided by the GIS, such as Feature, Superspan, Display_Rule, Control_Point, and others. In addition, GIS application and database designers can define their own object views for modelling different kinds of features, tables, and relationships to be maintained by GIS. As mentioned previously, Parcel is an example of a user-defined object view, added to the GIS object views via Nucleus Definition statements.

The various GIS and user-defined object views are related, in that each object view is related to a 'higher level' or more general object view (with the exception of the GIS object view, which is the 'highest' level view). For instance, the user-defined Parcel object view is a classification of the GIS Area Feature object view, which is a classification of the GIS Feature object view. The relationships between all GIS and user-defined object views are explained in "Overview of GIS Object Views". The list of Object View names in the GIS defines the list of names that can be the subject of a From clause in a GIS query (e.g., Select Owner From Parcel). This is similar to identifying all the view names available for manipulating data in a relational database. Once an user or application developer knows the names of these views, queries can be created to select data from the views. Object Views also share the same types of restrictions as relational views, in that relational views can be defined that restrict a user's ability to insert or update data. Likewise, there are GIS object views that are not valid for these operations. All object views are valid for data selection or retrieval (subject to GIS and corporate data security enforcement). Object View Components and Attributes All Object Views in GIS have components and attributes. Components of an Object View can be thought of as complicated attributes (they do not map to a single column of data), whereas an Attribute can be thought of as a 'simple' component (it maps to a single column of data). An object view component can be further broken down into a set, or sets, of attributes.

Projections Once the list of object view names has been identified, it is possible to use the '*' notation to select information from any object view. Once the list of object view components and attributes have been identified, it is possible to select specific sets of values from one or more object views. For instance, once it has been identified that the object view 'Parcel' has two attributes 'Owner' and 'Appraised_Value', users can construct the following queries: Select * From Parcel Where Owner = 'John Smith' Select Owner, Appraised_Value From Parcel Where Owner = 'John Smith' In the GIS, the specification of which component and attribute values are to be returned from a Select operation is called the Projection. An asterisk (*) is used as the notation for the attribute projection of all components and attributes of an object view, just as in SQL, 'Select * from Viewname' will return arrays of values for all columns of the view Viewname. Attribute Evaluation In GIS, any attribute value associated with an instance of data can be evaluated in a where clause. OVERVIEW OF GIS OBJECT VIEWS This section illustrates the relationships between all GIS object views. GIS Object View All data in a GIS database can be accessed via the GIS object view. The components and attributes of the GIS object view are those common to all GIS data. The GIS object view is further classified by the following object views: o Nucleus Object View o Library Object View o Administrative Object View There are two important concepts regarding the set of object views that 'further classify' a particular object view: o Any instance of data that can be accessed via a particular object view can only be accessed via one of the object views that further classify that view. In other words, the list of object views that classify a particular object view are 'mutually exclusive' with regard to instances of data. o The components and attributes of an object view can be treated as components and attributes of any object view that further classifies that view. The GIS object view is illustrated in Fig. 1. Fig. 1 GIS Object View. The GIS Object View is further classified by the three object views as shown in the figure: Administrative, Nucleus, and Library. Put simply, this means that any instance of data that can be accessed using the GIS object view is either nucleus, administrative, or library data. Additional GIS object views are provided to further classify data in the nucleus, administrative, and library object views as indicated by the dotted arrows. Administrative Object View Administrative data is a subset of GIS data that contains information which deals with the creation and management of the collection of DBMS tables that make up the GIS database. This includes definitions for the GIS system and user-defined object views. The "Administrative" object view is further classified by the object views listed in Fig. 2. Administrative object views exclude all library and nucleus data. If an application developer wanted to select all administrative data, the "Administrative" object view would be used. If the application developer wanted to select only administrative user profile data, the "User" object view could be used. Fig. 2 Administrative Object View. The object views that belong to the set of administrative object views are illustrated above. All instances of administrative data belong to one of these object views. There are no further object views or further classifications of object views for administrative data. Library Object View Library Objects consist of various rules and parameters that control how the user interacts with the system, how information is captured, how information is graphically and textually realized, and the meaning of units and projections. In short, this information defines much of the operation of a GIS and may be specific to individual users of groups of users or may be generic across an entire enterprise. The "Library" object view is further classified by the object views listed in Fig. 3. Library object views exclude all nucleus and administrative data. If an application developer wanted to select all library data, the "Library" object view would be used. If an application developer wanted to select only Macros, the "Macro" object view could be used.

Fig. 3 Library Object View. The object views that belong to the set of library object views are illustrated above. All instances of library data belong to one of these object views. There are no further object views or further classifications of object views for library data. Nucleus Object View Nucleus data is a subset of GIS data that contains geographic data and attributes associated with that data and is the subset of GIS data used to model and maintain an enterprise's geographic assets.

The following characteristics distinguish nucleus data from administrative or library data in GIS: o Nucleus data can be evaluated based on spatial (or location) characteristics. o Nucleus data can be manipulated by check-in, check-out, load-for-update and extract-updates operations. o The Nucleus object views can be extended by the database designer or application developer through GIS Nucleus definitions. Nucleus definitions permit the creation of user-defined object views for modelling features, relationships, and user-tables. Nucleus object views exclude all library and administrative data. If an application developer wanted to select all nucleus data, then the "Nucleus" object view would be used. If an application developer wanted to select only Features, then the "Feature" object view would be used. The Feature, Relationship and User_Table object views are further classified by additional GIS and user-defined object views. These object views are described in the following sections: Feature Object View The following characteristics distinguish feature data from other data in GIS: o Features can be associated with Display Rules.

Features have the following location characteristics: o Absolute Point o Node o Extent o Feature_Path The "Feature" object views exclude all library data and administrative data and all nucleus data such as user tables, boundaries, polygon sets, etc.. If an application developer wanted to select all feature data, then the "Feature" object view would be used. If the application developer wanted to select only point features, the "Point" object view would be used. If the application developer wanted to select only those point features defined as poles having a feature class of "telephone_pole", then the "telephone_pole" object view would be used. Each of the Feature object views may be further classified by user-defined feature names, such as 'Parcel' or 'Highway', that are created via GIS Nucleus Definitions. Nucleus definitions provide a means of associating user-defined attributes, such as 'Owner' with the Parcel object view. Each object view created via Feature definitions may be further classified, if desired, by specifying one or more Feature Classes within the Feature Definition. For instance, the enterprise may choose to further classify 'Parcel' features by defining 'Rural_Lot', 'Municipal_Lot' feature classes. Any user-defined attributes associated with Parcel data are also associated with all feature classes of the Parcel data; however, different Display Rules may be associated with the various feature classes for

Parcel data (e.g. 'Rural_Lot' features can be rendered differently than 'Municipal_Lot' features).

If an application developer or database designer creates a feature definition for an Area feature named Parcel, having a user-defined attribute named 'Owner', and feature classes 'Rural_Lot', 'Municipal_Lot', the following queries could be used to manipulate Parcel data: Select * from Parcel where Owner = 'John Smith' Select * from Rural_Lot where Owner = 'John Smith' Select * from Municipal_Lot where Owner = 'John Smith' The first query will return all instances of parcels owned by 'John Smith', the second will return only those instances of parcels that are classified as rural lots, the third, only municipal lots.

User_Table Object View User tables provide the ability to model non-geographic data (sets of attributes) as part of the GIS nucleus. The instances of user table data can be related to features via GIS user-defined relationships, and the user table data can be manipulated with checkin, checkout, load and extract operations. User tables are defined via GIS Nucleus definitions. Each User_Table name, when defined by the application developer or database designer, becomes an object view which further classifies the User_Table object view.

Relationship Object View The Relationship object view is further classified by GIS system maintained relationships, such as Layer_assoc and Theme_assoc. Application developers and database designers can create user-defined relationships via GIS Nucleus definitions. GIS provides a means of creating user-defined relationships. Each user-defined relationship name, when defined by the application developer or database designer, becomes an object view which further classifies the Relationship object view. As with features and user tables, the GIS nucleus definitions provide a means of associating sets of (user-defined) attributes with user-defined relationships. Object View Considerations for Applications and Data Access Recall that the list of GIS Object Views defines the 'names' that can be the subject of a From clause to access GIS data. For instance, using the Object Views listed thus far, the following are valid queries: A) Select * From GIS B) Select * From Nucleus C) Select * From Feature D) Select * From Span If a GIS database designer creates a Feature Definition for 'Highway', having a Feature Class called 'Interstate', then the following two queries would also be valid:

E) Select * From Highway F) Select * From Interstate The relationship between these object views is much like a relationship between various views on the same set of relational tables. The object view specified in the From clause defines the scope of the data that can possibly be returned from the query. The object view is also a consideration in performance of queries, since the more general object views will generally indicate searches through a wider range of data. Conclusion This disclosure presented a Relational Data Model for a Geographic Information System that supports inheritance. The inheritance model is superior to conventional relational models, because it recognizes the inheritance relationships that exist in GIS. An implementation of the inheritance model was proposed using relational technology. A relational approach has several advantages over object-oriented implementations. First, relational DBMSs are commonplace in the business world. Second, relational systems support application and data independence through abstraction and views. Third, distributed relational systems are reaching maturity. An actual GIS data model was presented. This model provided support for the three major areas in a GIS: spatial data, application development, and administrative tasks.

Previous Doc Next Doc Go to Doc#